第一章

一. 1. 管理计算机硬件与软件资源的核心系统软件，为应用程序提供运行环境，同时作为用户与计算机硬件之间的交互接口，协调硬件资源的使用并保障系统安全稳定运行。

4. 将时间作为关键参数，系统能及时响应外部事件的请求。

5. 系统中的某些资源一段时间内只允许一个进程访问资源

三. 1. 硬件 软件　　3. 硬件资源 软件资源

5. 共享 异步

第二章

一. 3. 进程中访问共享资源的那段代码。

4. 用于协调多个进程对共享资源访问的机制

三. 1. 动态性 异步性　　5. 动态 静态

6. 互斥制约。　　8. 管道 消息传递 共享内存

9. 被执行 就绪 阻塞

五. 2. S1无前驱，S2前驱是S1，S3前驱是S1，S4前驱是S2，S5前驱是S2和S3，S6前驱是S3，S7前驱是S4，S5和S6。

信号量机制设计：

```
Semaphore sasemaphore_S2 = 0, semaphore_S3=0 ..... semaphore_S7=0;
void S1 (){
        V(semaphore_S2);
        V(semaphore_S3); }
void S2 (){
        P(semaphore_S2);
        V(semaphore_S4);
        V(semaphore_S5); }
void S3 (){
        P(semaphore_S3);
        V(semaphore_S5);
        V(semaphore_S6); }
```

```
void S4 (){
        P(semaphore_S4);
        V(semaphore_S7); }
void S5 (){
        P(semaphore_S5);
        V(semaphore_S7); }
void S6 (){
        P(semaphore_S6);
        V(semaphore_S7); }
void S7 (){
        P(semaphore_S7); }
```

3.

semaphore empty=5, orange=0, apple=0, mutex=1;

~~Dad(){while(1){wait(empty);~~

```
Dad(){
    while(1){
        wait(empty);
        wait(mutex);
        将水果放入盘中;
        signal(mutex);
        if(放入的是橘子)
          signal(orange);
          else signal(apple);
        }
    }
```

```
Son(){
    while(1){
      wait(orange);
      wait(mutex);
      从盘中拿水果(橘子);
      signal(mutex);
      signal(empty);
      享用橘子;
    }
}
```

```
Daughter(){
    while(1){
      wait(apple);
      wait(mutex);
      从盘中取一个苹果;
      signal(mutex);
      signal(empty);
      享用苹果;
    }
}
```

4. semaphore empty1, ~~full1~~ full1, empty2, full2;

```
empty1 = 1;
full1 = 0;
empty2 = 1;
full2 = 0;
main(){
    cobegin{
        PA();
        PB();
        PC();
    }
    coend;
}
```

```
PA(){
    while(1){
      从磁盘读一记录;
      P(empty1);
      存入缓冲区1;
      V(full1);
    }
}
```

```
PB(){
    while(1){
      P(full1);
      从缓冲区1取出记录;
      V(empty1);
      P(empty2);
      将记录存入缓冲区2;
      V(full2);
    }
}
```

```
PC(){
    while(1){
      P(full2);
      从缓冲区2取出记录;
      V(empty2);
      打印记录;
    }
}
```

5.

```
var
  count: integer:=0;
  mutex, sofa, empty, full: semaphore:=1, N, 1, 0;
  cut, payment, receipt, semaphore:=0, 0, 0;

begin
  parbegin
    guest: begin
      wait(mutex);
      if(count>N) then begin
        signal(mutex);
        exit shop;
      end else begin
        count:=count+1;
      if(count>1) then begin
        signal(mutex);
        wait(sofa);
        sit on sofa;
        wait(empty);
      sit on the barber-chair;
      signal(full);
      wait(cut);
      get up from the barber-chair;
      wait(payment);
      pay;
      signal ~~(receipt)~~(receipt);
      ~~end~~
    end
      wait(mutex);
      count:=count-1;
      signal(mutex);
      exit shop;
      end
    end.
    barber: begin
      repeat
        wait(full);
        signal(empty);
        cut hair;
        ~~signal(cut)~~
        signal(cut);
        wait(payment);
        accept payment;
        signal(receipt);
      until false;
      end
    parend.
  end.
```

第三章

一. 2. 合理分配处理机资源.

3. 操作系统中衡量进程执行效率的重要指标之一. 周转时间=进程完成时间-进程提交时间.

4. 死锁: 两个或多个进程因相互等待对方持有的资源而陷入无限等待的状态.

三. 2. 提交 后备 完成　　6. 预防 避免 检测 解除

五. 1. ① FCFS算法　完成时间: A: 0+3=3 B: 3+6=9 C: 9+4=13 D: 13+5=18 E: 18+2=20.

周转时间: A: 3-0=3 B: 9-2=7 C: 13-4=9 D: 18-6=12 E: 20-8=12

带权周转时间: A: $\frac{3}{3}=1$ B: $\frac{7}{6}$ C: $\frac{9}{4}=2.25$ D: $\frac{12}{5}=2.4$ E: $\frac{12}{2}=6$.

平均周转时间: (3+7+9+12+12)÷5=9.4

平均带权周转时间 $(1+\frac{7}{6}+2.25+2.4+6)\div 5\approx 2.567$.

② 非抢占SJF算法. 完成时间 A: 0+3=3 BC: 3+4=7 E: 7+2=9 D: 9+5=14 B: 14+6=20.

周转时间: A: 3-0=3 B: 20-2=18 C: 7-4=3 D: 14-6=8 E: 9-8=1

带权周转时间: A: $\frac{3}{3}=1$ B: $\frac{18}{6}=3$ C $\frac{3}{4}=0.75$ D: $\frac{8}{5}=1.6$ E: $\frac{1}{2}=0.5$

平均周转时间: (3+18+3+8+1)÷5=7.4

平均带权周转时间 (1+3+0.75+1.6+0.5)÷5=1.55.

③ 抢占SJF算法. 完成时间 A: 0+3=3 C: 3+4=7 E: 7+2=9 D: 9+5=14 B: 14+6=20.

周转时间: A: 3-0=3 B: 20-2=18 C: 7-4=3 D: 14-6=8 E: 9-8=1

带权周转时间: A: $\frac{3}{3}=1$ B: $\frac{18}{6}=3$ C: $\frac{3}{4}=0.75$ D: $\frac{8}{5}=1.6$ E: $\frac{1}{2}=0.5$

平均周转时间: (3+18+3+8+1)÷5=7.4

平均带权周转时间: (1+3+0.75+1.6+0.5)÷5=1.55.

④ HRRN算法　完成时间: A: 0+3=3 C: 3+4=7 E: 7+2=9 D: 9+5=14

周转时间: A=3-0=3 B=9-2=7 C=13-4=9 E=15-8=7 D=14-6=14

带权周转时间: A: $\frac{3}{3}=1$ B: $\frac{7}{6}\approx 1.17$ C: $\frac{9}{4}=2.25$ E: $\frac{7}{2}=3.5$ D: $\frac{14}{5}=2.8$

平均周转时间: (3+7+9+14+7)÷5=8

平均带权周转时间: (1+1.17+2.25+2.8+3.5)÷5=2.144.

⑤ RR算法.

周转时间: A: 5-0=5 B: 11-2=9 C: 12-4=8 E: 16-8=8 D: 20-6=14

带权周转时间: A: $\frac{5}{3}\approx 1.67$ B: $\frac{9}{6}=1.5$ C: $\frac{8}{4}=2$ E: $\frac{8}{2}=4$ D: $14\div 5=2.8$

平均周转时间: (5+9+8+14+8)÷5=8.8

平均带权周转时间: (1.67+1.5+2+2.8+4)÷5=2.394

2.

所有任务的松弛度：

A：20ms-0ms-10ms=10ms　　B：50ms-0ms-10ms=40ms　　C：50ms-0ms-15ms=35ms

由于任务A松弛度最低，先执行任务A.

重新计算B、C松弛度.

B：50ms-10ms-10ms=30ms　　C：50ms-10ms-15ms=25ms

任务C松弛度最小，接下来执行任务C.

再次计算

A：20ms-20ms-10ms=-10ms　　B：50ms-35ms-10ms=5ms

任务B拥有最低的正松弛度，因此执行任务B.

综上，调度的顺序是任务A→任务C→任务B.

3. (1) Allocation = MAX - Need 得

Process	Allocation
P0	0 0 3 2
P1	1 0 0 0
P2	1 3 5 4
P3	0 3 3 2
P4	0 0 1 4

系统是安全的.

(2) 系统按银行家算法进行检查.

① $Request_2(1,2,2,2) \le Need_2(2,3,5,6)$

② $Request_2(1,2,2,2) \le Available(1,6,2,2)$

③ 系统先假定为P2分配资源，并修改 Available, $Allocation_2$ 和 $Need_2$ 向量

$Available = (0,4,0,0)$

$Allocation_2 = (2,5,7,6)$

$Need_2 = (1,1,3,4)$

④ 进行安全检查：此时对所有的进程，条件 $Need_i \le Available\ (0,4,0,0)$ 都不成立.

即Available不能满足任何进程的请求，故系统进入不安全状态.

因此，当进程P2提出请求 Request (1,2,2,2) 后，系统不能将资源分配给它.

(3) 系统立即满足进程P2的请求 (1,2,2,2) 后，并没有马上进入死锁状态.

因为此时上述进程并没有申请新的资源。